通過テスト９ 解答

9 オーバーロードとオーバーライド

1. B, C, E

2.

```
package e2;
class  Tools {
    public void clear(int[] a){
        for(int i=0; i<a.length; i++){
            a[i] = 0;
        }
    }
    public boolean isEmpty(int n){
        if(n==0)return true;
        return   false;
    }
}
class MyTools extends Tools {
    public void clear(int[][] n){
        for(int i=0; i<n.length; i++){
            clear(n[i]);
        }
    }
    public void clear(int[][][] n){
        for(int i=0; i<n.length; i++){
            clear(n[i]);
        }
    }
    public boolean isEmpty(String s){
        if(s==null)         return true;
        if(s.equals(""))return true;
        return   false;
    }
}
```

```
public class Exec {
    public static void main(String[] args) {
        int[][][] a ={ {{10,20},{30,40}},{{50,60,70}}};
        String   s = "";
        int   n = 10;
        MyTools t = new MyTools();
        //
        t.clear(a);
        for(int[][] n1 : a){
            for(int[] n2 : n1){
              for(int n3 : n2){
                  System.out.print(n3+ " ");
              }
              System.out.println("");
            }
        }
        System.out.println(t.isEmpty(s));
        System.out.println(t.isEmpty(n));
    }
}
```

【解説】1 次元配列をクリアするクリアメソッドを利用すると，2 次元配列をクリアするメソッドが簡単に書ける．2 次元配列から要素をひとつ取り出し，1 次元配列をクリアするメソッドを使ってそれをクリアする．また，3 次元配列をクリアするメソッドも同様に 2 次元配列をクリアするメソッドを利用すると，要素をひとつ取り出し，2 次元配列をクリアするメソッドを使ってクリアできる．

3.

```
package e3;
class Color {
    public void paint(){
        System.out.println("なし");
    }
}
class Blue extends Color{
    public void paint(){
        System.out.println("青");
    }
}
class Red extends Color{
    public void paint(){
        System.out.println("赤");
    }
}
public class Exec {
    public static void main(String[] args) {
        Color   c   =   new Color();
        Blue    b   =   new Blue();
        Red     r   =   new Red();
        c.paint();
        b.paint();
        r.paint();
    }
}
```

4.　A

【解説】16 行目では Version3 型のオブジェクトを作成して Version2 型の変数に代入しています．17 行目の v2.setVn(10)では，12 行目の setVn メソッドが起動します（オーバーライド）．12 行目ではフィールド vn に値をセットしますが，この Vn は 11 行目に宣言している Version3 クラスのフィールド変数です．同じ vn が Version1 クラスにもありますが，サブクラス側が優先されます．最後に v2.disp()を実行しますが，disp メソッドは 8 行目の Version2 クラスのメソッドが起動します．この disp メソッドは vn を表示しますが，アクセスする Vn は 2 行目にある Version1 クラスの vn で，これは初期値のままになっているので，0 が表示されます．

5.　A, C, E, G

【解説】A は正しいオーバーライド．B は戻り値型を変えた不正なオーバーライド．C は引数が違うのでオーバーロード.D はオーバーライドだが，戻り値として Dog 型を Tom 型にキャストできないのでコンパイルエラー．E は共変戻り値を使ったオーバーライド.F はス

ーパークラスである Object を返すので共変戻り値にならず不正. G は引数が違うので正しいオーバーロード.

6.　＜MyLogin クラス＞

```
package e6;
import lib.Input;
class MyLogin extends Login{

    // tanaka のパスワードは tnk、yamada のパスワードは ymd、のように同じ並び順で定義する
    String[] passWord  = {"tnk", "ymd"};
    String   pw;

    // ユーザー名とパスワードを入力するようにオーバーライドする
    public void getInfo(){
        uid = Input.getString("ユーザー名");
        pw  = Input.getString("パスワード");
    }
    // ユーザー名とパスワードの両方がマッチすればユーザーとみなすようにオーバーライドする
    public boolean isUser(){
        for(int i=0; i<passWord.length; i++){
            if(pw.equals(passWord[i]) && uid.equals(idList[i])){
              return true;
            }
        }
        return false;
    }
}
```

＜Exec2 クラス＞

```
package e6;
public class Exec2 {
    public static void main(String[] args) {
        Auth  au  =  new Auth();
        MyLogin mlo =   new MyLogin();
        au.check(mlo);
    }
}
```

【解説】Login クラスを継承して getInfo メソッドと isUser メソッドをオーバーライドすることがポイントである. Auth クラスは一切書き換える必要がない. check メソッドの引数にに Login 型の参照を指定する代わりに MyLogin 型の参照を指定してよい. ポリモーフィズムにより Auth クラスではオーバーライドされた getInfo と isUser が実行される.